## 软件编码规范

## 类文档注释（适用于C/C++）

在类定义前面应加上类说明文档。

例如：

/\*\* WindowsNT

\* @brief Windows Nice Try.

\* @author Bill Gates

\* @author Several species of small furry animals gathered together

\* in a cave and grooving with a pict.

\* @version 4.0

\* @date 1996-1998

\* @bug It crashes a lot and requires huge amounts of memory.

\* @bug The class introduces the more bugs, the longer it is used.

\* @warning This class may explode in your face.

\* @warning If you inherit anything from this class, you're doomed.

\*/

class WindowsNT {};

## 语句块注释（适用于C/C++）

* 语句块的注释可以用在语句块的开头和结束位置：

{

// Block1 (meaningful comment about Block1)

... some code

{

// Block2 (meaningful comment about Block2)

... some code

} // End Block2

} // End Block1

## 不要用if语句的默认方法测试非零值（适用于C/C++）

* If语句只用于检测布尔值(bool)，不要用默认的方法测试非零值，比如：

建议使用：

if (FAIL != f())

不建议使用下面的表达式：

if (f())

* 宏定义的情况也一样：

#define STREQ(a, b) (strcmp((a), (b)) == 0)

或者使用内联函数：

inline bool

StringEqual(char\* a, char\* b)

{

(strcmp(a, b) == 0) ? return true : return false;

Or more compactly:

return strcmp(a, b) == 0;

}

## 布尔类型 （适用于C/C++）

* 早期的C++没有布尔类型，但新的C++标准增加了布尔类型。如果可以使用内置的布尔类型的情况下，应使用布尔类型。

早期的布尔类型定义为：

typedef int bool;

#define TRUE 1

#define FALSE 0

或：

const int TRUE = 1;

const int FALSE = 0;

* 在这种情况下，条件表达式不要比较1值(如TRUE，YES等等)，而要用0值(如FALSE,NO等等)进行比较。因为多数函数返回0表示FALSE，而非零表示TRUE。如：

if (TRUE == func()) { ... // 错误：假如func()返回 2 怎么办？

必须写成：

if (FALSE != func()) { ...

## 不要直接使用数字 （适用于C/C++）

* 直接使用数字，会使源代码难以理解和维护。如：

if (22 == foo) { start\_thermo\_nuclear\_war(); }

else if (19 == foo) { refund\_lotso\_money(); }

else if (16 == foo) { infinite\_loop(); }

else { cry\_cause\_im\_lost(); }

当一段时间过去以后，有谁会记得22和19是什么意思？假如数字改变，或者是编写错误，更是难以发现问题。

* 我们可以用#define或者常量来改变这一状况，如：

#define PRESIDENT\_WENT\_CRAZY (22)

const int WE\_GOOFED= 19;

enum

{

THEY\_DIDNT\_PAY= 16

};

if (PRESIDENT\_WENT\_CRAZY == foo) { start\_thermo\_nuclear\_war(); }

else if (WE\_GOOFED == foo) { refund\_lotso\_money(); }

else if (THEY\_DIDNT\_PAY == foo) { infinite\_loop(); }

else { happy\_days\_i\_know\_why\_im\_here(); }

## 指针变量的命名前缀（适用于C/C++）

* 指针变量多数情况应在前面加 'p'。
* 星号 '\*' 应靠近类型，而不是变量名。

例如：

String\* pName= new String;

特别的：String\* pName, name; 应分成两行来写：

String\* pName;

String name;

## 引用变量和返回引用函数的命名前缀（适用于C++）

* 引用必须用 'r'作前缀修饰。

例如：

class Test

{

public:

void DoSomething(StatusInfo& rStatus);

StatusInfo& rStatus();

const StatusInfo& Status() const; // 这里返回的是常量引用，所以不符合本规则

private:

StatusInfo& mrStatus;

}

## 静态变量的命名前缀（适用于C++）

* 静态变量以 's' 作为前缀。

例如：

class Test

{

public:

private:

static StatusInfo msStatus;

## C 函数的命名（适用于C/C++）

* C++项目中，应尽量少用C函数。
* C函数使用GNU规范，所有字母都使用小写，并用下划线 '\_' 作为单词的分隔。

例如：

int

some\_bloody\_function()

{

}

* 特别的，为了赚容C/C++，在必要的时候，在C++中应以下面的格式定义C函数：

extern “C” int some\_bloody\_function();

* 或在C/C++中推荐使用下面的格式：

#ifdef \_\_cplusplus\_\_

extern “C” {

#endif

int

some\_bloody\_function()

{

}

#ifdef \_\_cplusplus\_\_

}

#endif